Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Katanec

Zajemanje podatkovnih paketov med elektroniko gospodinjskega aparata in WiFi modulom

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in matematika

Mentor: prof. dr. Nikolaj Zimic

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Z razmahom interneta stvari se v internet vse pogosteje vključujejo tudi aparati bele tehnike, za kar je potrebno izdelati vmesnik aparata, tako strojni kot programski, in poskrbeti za ustrezno odpravljanje napak. V prvi fazi razvoja se običajno obstoječa elektronika aparata poveže z WiFi komunikacijskim modulom preko serijskih vrat. Na ta način se skrajša čas razvoja in hitro pride do delujočega prototipa, ki omogoča povezavo v internet. V diplomski nalogi izdelajte program, ki bo omogočal zajemanje podatkov na serijskih vratih med elektroniko aparata in WiFi vmesnikom. Program naj omogoča človeku prijazen izpis zajetih paketov ter samodejno iskanje nekaterih napak protokola. Uporabniški vmesnik naj bo zgrajen tako, da bo razvojnemu inženirju omogočal enostavno kontrolo podatkovne komunikacije ter analizo prejetih paketov.

Zahvaljujem se mentorju prof. dr. Nikolaju Zimicu za vse napotke pri diplomskem delu. Iskrena hvala podjetju Gorenje d. d., ki mi je omogočilo vso potrebno opremo za izvedbo praktičnega dela naloge. Zahvala gre delovnemu mentorju Juretu Ernstu za usmeritve pri razvoju aplikacije v okviru diplomske naloge. Hvala prof. Darji Skutnik in prof. Evi Meža za lektoriranje. Iskrena hvala tudi vsem bližnjim za podporo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
XMC		Opisni jezik
UART	Universal asynchronous receiver/transmitter	Univerzalni asinhroni sprejemnik/oddajnik
XML	Extensible Markup Language	Razširljiv označevalni jezik
XAML	Extensible Application Markup Language	Razširljiv označevalni jezik za aplikacije
IoT	Internet of Things	Internet stvari
ECHO	Electronic Computing Home Operator	Elektronski računalniški domači operater
BLE	Bluetooth Low Energy	Nizko potratni Bluetooth
IP	Internet Protocol	Internetni protokol
IPv6	Internet Protocol version 6	Internetni protokol verzije 6
6LowPAN	IPv6 Low-power wireless Personal Area Networks	IPv6 preko nizko energijskega brezžičnega osebnega omrežja
JSON	JavaScript Object Notation	JavaScript objektna notacija
TLS	Transport Layer Security	Plast, ki skrbi za varen prenos podatkov
SSL	Secure Sockets Layer	Kriptigrafski protokol za varno plast vtičnic

WPA2	WiFi Protected Access II	Protokol za varen dostop do WiFi
DTE	Data Terminal Equipment	Podatkovna terminalska oprema
DCE	Data Communication Equipment	Oprema za podatkovno komunikacijo
TXD	Transmit Data	Pošiljanje podatkov
RXD	Receive Data	Prejem podatkov
GND	Common Ground	Skupna masa
MSB	Most Significant Bit	Najpomembnejši bit
LSB	Least Significant Bit	Najmanj pomemben bit
RTS	Request-to-Send	Zahteva za pošiljanje
CTS	Clear-to-Send	Prosto za pošiljanje
ACK	Acknowledgement	Potrditev
NACK	Negative Acknowledgement	Negativna potrditev
SOF	Start of Frame	Začetek okvirja
EOF	End of Frame	Konec okvirja
FN	Frame Number	Številka okvirja
WPF	Windows Presentation Foundation	Windowsova platforma za izdelavo uporabniškega vmesnika
MVC	Model–View–Controller	Model - pogled - krmilnik
MVVM	Model–View–ViewModel	Model - pogled - pogled modela
SQL	Structured Query Language	Strukturirani poizvedovalni jezik
USB	Universal Serial Bus	Vodilo za priklop perifernih naprav na računalnik

Povzetek

Naslov: Zajemanje podatkovnih paketov med elektroniko gospodinjskega aparata in WiFi modulom

Avtor: Jernej Katanec

Vse več gospodinjskih aparatov se povezuje v omrežje. Trend razvoja gre v tej smeri, da jih lahko oddaljeno nadzorujemo in upravljamo s pomočjo mobilne aplikacije. Pri komunikaciji naprav med seboj je prva faza komunikacija med WiFi modulom in elektroniko gospodinjskega aparata. V okviru te diplomske naloge smo se povezali na povezavo med njima. Cilj je bil razviti aplikacijo, s katero lahko prestregamo komunikacijske pakete, te pa potem v uporabniku prijazni obliki izpišemo na zaslon. Ker je ta komunikacija zelo intenzivna, smo dodali tudi opcijo, da lahko uporabnik pakete filtrira po različnih ključih. Aplikacija se bo v praksi uporabljala za nadzor nad to fazo komunikacije in za diagnosticiranje napak.

Ključne besede: WiFi, hišna avtomatizacija, WiFi modul, gospodinjski aparati, prestrezanje dvosmerne komunikacije, komunikacijski protokoli, C#, XAML, MVVM.

Abstract

Title: The Capturing of Data Packets Between the Electronics of Household Appliance And the WiFi Module

Author: Jernej Katanec

More and more household appliances are connected to a network. The development trend for them is to be controlled remotely and managed through a mobile application. The first phase in device communication is communication between the WiFi module and the electronics of the household appliance. Within this thesis we have connected our computer to the connection between the two. The aim was to develop application that allows us to intercept communication packages, and after that it displays them on the screen in a user-friendly form. Since this communication is very intensive, we have added an option that allows the user to filter the packages by different keys. In practice, the application will be used to control this phase of communication and to diagnose errors.

Keywords: WiFi, home automation, WiFi module, household appliances, interception of two-way communication, communication protocols, C#, XAML, MVVM.

Poglavje 1

Uvod

V sodobnem času pametnih naprav je zahteva po povezljivosti oz. komunikaciji med njimi vedno večja. Gospodinjski aparati niso nobena izjema. S svojim telefonom želimo na daljavo nastaviti temperaturo hladilnika, dobiti obvestilo ob koncu cikla pranja pralnega stroja ali zmanjšati moč kuhinjske plošče ob vretju vode. Gospodinjske naprave pa lahko komunicirajo tudi med sabo - na primer, ko prižgemo kuhalno ploščo, ta pošlje signal kuhinjski napi, naj se vklopi.

V tej diplomski nalogi se bomo najprej dotaknili ozadja delovanja komunikacije med povezljivimi gospodinjskimi aparati. Pogledali bomo, kaj je hišna avtomatizacija, dotaknili se bomo njene zgodovine in raziskali, katera tehnologija se pri tem največkrat uporablja. Pregledali bomo tudi ponudbo na trgu, kjer bo poudarek na povezljivih gospodinjskih aparatih.

V nadaljevanju se bomo osredotočili na prvi del komunikacije, ki poteka med elektroniko gospodinjskega aparata in WiFi modulom, ki je z njo fizično povezan preko RS-232 povezave. Z računalnikom smo se povezali na ta del komunikacije in prestrezali pakete protokola Inspector. Glavni cilj je bil razviti aplikacijo, ki bo prestrežene pakete pretvorila v uporabniku prijazno obliko, ta pa bo z njeno pomočjo lahko nadzoroval promet med elektroniko gospodinjskega aparata in WiFi modulom. Uporabnik lahko s to aplikacijo prestrežene pakete analizira in diagnosticira določene napake.

V tretjem poglavju si bomo pogledali, kako smo vzpostavili naše delovno okolje, nekaj lastnosti WiFi modulov ter elektronike gospodinjskih aparatov, na kratko pa bomo opisali standard RS-232. Predstavili bomo tudi protokola Inspector in XMC, ki ju obravnavamo pri prestrezanju paketov, ter tehnologijo, ki smo jo uporabili pri razvoju aplikacije.

V četrtem poglavju bomo podrobno predstavili funkcionalnosti aplikacije ter se dotaknili glavnih točk pri njenem razvoju. Za konec bomo predstavili še možne izboljšave aplikacije.

Poglavje 2

Hišna avtomatizacija in povezljivi gospodinjski aparati

Hišna avtomatizacija oz. "domotika"(ang. *domotics*) je avtomatiziranje doma, ki mu nato tudi rečemo "pameten dom"ali "pametna hiša". Izraz domotika izhaja iz latinske besede "domus", kar pomeni dom, in francoske "informatique", informacija [1]. Zajema avtomatizacijo in nadzor luči, ogrevanja, ventilacije, klimatiziranja, varnosti in tudi gospodinjskih aparatov, kot so pralni/sušilni stroji, pečice, kuhalne plošče, hladilniki, zamrzovalniki in podobno. Moderni sistemi so v glavnem sestavljeni iz stikal in senzorjev, ki so povezani na centralno bazo. Od tu lahko uporabniki upravljajo z napravami preko uporabniškega vmesnika. Ta je največkrat v obliki terminala, vgrajenega na steno, ali aplikacije na mobilni napravi, včasih pa tudi kot spletna aplikacija preko internetnih storitev v oblaku. Domače naprave, ki jih oddaljeno upravljamo in nadzorujemo preko interneta, so pomemben del "interneta stvari"(ang. *Internet of Things - IoT*) [11].

V tej diplomski nalogi se bomo osredotočili predvsem na povezljive gospodinjske aparate oziroma bolj specifično na belo tehniko. Pred leti je trend razvoja pametnih naprav bele tehnike stremel k povezovanju samih naprav med sabo (na primer kuhinjske plošče ter nape) in združevanju njihovih ključnih funkcij. Zaradi problemov s pomanjkanjem nadzora uporabnika nad dogaja-

njem so ta princip večinoma opustili. Večina večjih proizvajalcev bele tehnike danes že uporablja strojno opremo, ki omogoča, da gospodinjske aparate nadzorujemo in upravljamo z aplikacijo na mobilni napravi.

Svetovni trg iz področja hišne avtomatizacije je bil po podatkih družbe *Research and Markets* leta 2013 vreden 5,77 milijarde dolarjev, ocenjujejo pa, da naj bi se ta številka do leta 2020 povzpela na 12,81 milijarde ameriških dolarjev [18].

2.1 Zgodovina

S konceptom oddaljenega upravljanja se je ukvarjal že Nikola Tesla, ki je leta 1898 patentiral izum vodnega plovila, ki ga je bilo moč upravljati na daljavo s pomočjo radijskih valov. Ideja takrat ni bila realizirana, saj večina strokovnjakov takrat še ni verjela, da radijski valovi resnično obstajajo [20].

Tri leta za tem se je z izumom prvega sesalca začel razvoj gospodinjskih aparatov. Začetek njihove komercialne prodaje sega v 20. leta prejšnjega stoletja, a so bili sprva dostopni le premožnejšim ljudem, ki so že imeli elektriko v svojih domovih. Večina električnih gospodinjskih aparatov, kot so hladilnik (1913), pomivalni (1929), pralni (1904) in sušilni stroj (1938), je bilo razvitih v prvi polovici 20. stoletja.

Prva pametna naprava, ki sicer nikoli ni bila komercialno prodana, je bila razvita že leta 1966. ECHO IV (Electronic Computing Home Operator) je bila naprava, ki jo je izumil Jim Sutherland, mnogi pa jo označujejo kot prvi domači računalnik. Med drugim naj bi znal računati nakupovalne liste, kontrolirati temperaturo doma ter prižigati in ugašati gospodinjske aparate [19].

Leta 1975 je škotsko podjetje Pico Electronics predstavilo projekt X10, ki je zagotavljal oddaljen nadzor gospodinjskih aparatov. To je bila prva tehnologija, ki je bila uporabljena v okviru hišne avtomatizacije. X10 komunicira med oddajnikom in sprejemnikom s pošiljanjem in prejemanjem signalov po žici.

Razcvet je hišna avtomatizacija doživela v prvih letih tega tisočletja. Pametne hiše so postale bolj dostopne, sprva pa je bil največji poudarek na varnostnih sistemih in varčevanju z energijo. Trenutni trendi hišne avtomatizacije vključujejo oddaljen mobilni nadzor, avtomatizirane luči in ogrevanje/klimatiziranje hiše, upravljanje z gospodinskimi aparati ter oddaljen video nadzor.

2.2 Ponudba na trgu

Trg danes ponuja ogromno različnih pametnih naprav, ki lahko komunicirajo med sabo. Mi se bomo v tem delu osredotočili predvsem na belo tehniko.

Eno izmed vodilnih podjetij na področju povezljivih gospodinjskih aparatov je ameriško podjetje General Electric (GE) [21]. Razvili so sistem *WiFi Connect*, ki omogoča, da z mobilnimi aplikacijami upravljamo z gospodinjskimi aparati tega podjetja. *WiFi Connect* se deli na tri dele - *Kitchen* (Kuhinja), ki zajema hladilnike, štedilnike, pečice in pomivalne stroje, *Laundry* (Perilo) vključuje pralne in pomivalne stroje ter *Comfort* (Udobje), ki zajema klimatske naprave in grelnike vode. Za vsakega od teh delov so razvili svojo mobilno aplikacijo. Vsaka aplikacija ima različne funkcije za posamezne naprave - npr. *WiFi Connect Kitchen* nam za hladilnik sporoči, ko je temperatura v hladilniku previsoka, ko mora biti zamenjan vodni filter ali ko pustimo odprta vrata hladilnika. Na daljavo lahko nastavimo temperaturo hladilnika, vključimo izdelovalca ledu ali pa vključimo gretje vode, kar omogočajo nekateri hladilniki podjetja GE [4].

WiFi Connect omogoča dva načina povezave:

- *Built-In Wifi Connect* (Vgrajen *WiFi Connect*) - v gospodinjski aparat je vgrajen WiFi modul, ki omogoča komunikacijo z mobilno napravo.
- *Optional WiFi Connect* (Opcijski *WiFi Connect*) - naprave, ki so kompatibilne GE *WiFi Connect*, ampak nimajo WiFi modula, lahko priključimo v omrežje s pomočjo *GE ConnectPlus* modula (slika 2.1).

Tega priključimo na komunikacijska vrata naprave in deluje kot navaden WiFi modul.

Slika 2.1: GE ConnectPlus [15].

Za oba načina potrebujemo v našem stanovanju WiFi usmerjevalnik, če želimo, da delujeta.

Nekateri evropski proizvajalci bele tehnike, kot so Bosch, Gaggenau, Neff in Siemens, za oddaljen nadzor in upravljanje uporabljajo tehnologijo *Home Connect*. Tudi ta deluje preko mobilne aplikacije. Za delovanje potrebujemo WiFi modul na gospodinjskem aparatu in domače brezžično omrežje. *Home Connect* je prva aplikacija, s katero lahko nadzorujemo in upravljamo z gospodinjskimi aparati različnih proizvajalcev, ki ustrezajo *Home Connect* standardu. Zaenkrat je na voljo v Nemčiji, Avstriji, na Nizozemskem, v Luksemburgu, Belgiji, Franciji, Švici in na Kitajskem [5].

Večina ostalih večjih proizvajalcev bele tehnike je razvila svoje lastne sisteme za oddaljen nadzor in upravljanje gospodinjskih aparatov. LG je razvil *SmartThinQ*, Samsung *SmartThings*, podoben sistem ima tudi Whirlpool. Vsi delujejo po enakem principu, torej so v omrežje povezani preko WiFi modula, na mobilne naprave pa si lahko naložimo aplikacijo, s katero nadzorujemo in upravljamo gospodinjske aparate.

Vodilni slovenski proizvajalec na področju bele tehnike, podjetje Gorenje, namerava pod lastno blagovno znamko svoje rešitve za povezanje naprav v omrežje predstaviti do konca tega leta. Lani pa so na sejmu elektronike IFA 2016 predstavili povezljive aparate pod svojo premijsko blagovno znamko

Asko. Njihovi aparati pametnega doma obsegajo povezljive pečice, hladilnike, kuhališča in nape ter seveda pomivalne stroje, pralne stroje, sušilne stroje in grelnike za vodo. Te aparate je moč upravljati in nadzorovati z mobilno aplikacijo [6].

2.3 Nadzor in upravljanje aparatov z glasovnimi ukazi

Amazon je razvil pametnega osebnega asistenta, ki se imenuje Alexa. Lahko prepoznava glasovne ukaze, predvaja glasbo ali zvočne knjige, upravlja alarme, zagotavlja novice v realnem času in pove podatke o vremenu ali prometu. Alexa lahko tudi kontrolira pametne naprave in je s tem uporabna kot sistem domače avtomatizacije. Pri podjetju GE so razvili zvočnega asistenta Geneva, ki prepoznava ukaze za njihove pametne gospodinjske aparate. Tega je na Alexi najprej potrebno aktivirati z glasovnim ukazom. Alexa deluje na napravah Amazon Echo (slika 2.2), Echo Dot, Echo Show in Amazon Fire TV [22] ali pa preko mobilne aplikacije Amazon. Na večini teh naprav se glasovno prepoznavanje aktivira z "budnico"(ang. wake-word), kot na primer *Echo* ali *Alexa*, na ostalih (npr. na mobilni aplikaciji Amazon) pa je potreben pritisk gumba. Zaenkrat je glasovno prepoznavanje implementirano le za angleški in nemški jezik [2].

Slika 2.2: Amazon Echo [16].

Google Assistant je pametni osebni asistent, ki so ga razvili pri Googlu, javnosti pa so ga prvi predstavili maja leta 2016. Za razliko od njegovega "starejšega brata", Google Now, lahko dialog z uporabnikom poteka v obeh smereh (medtem ko Google Now le postreže z rezultati želene poizvedbe). Google Assistant deluje na Google Home napravah (slika 2.3), na Pixel pametnih telefonih ter na nekaterih drugih mobilnih napravah z operacijskim sesitemom Android. Uporabniki lahko z njim komunicirajo z naravnim glasom, možna pa je tudi uporaba tipkovnice. Zvočni asistent podjetja GE Geneva deluje tudi na Google Assistant in lahko preko njega upravljamo in nadzorujemo gaspodinjske aparate [22] [3].

Slika 2.3: Google Home [17].

Tudi *Home Connect* lahko upravljamo preko Amazon Alexe, a zaenkrat zajema samo pečice, pralne stroje in avtomate za kavo. Google Assistant pri njih zaenkrat še ni podprt.

2.4 "Works with Nest"

Podjetje Nest Labs je razvilo program "Works with Nest"("Deluje z izdelki Nest"), ki omogoča, da njihove naprave komunicirajo z napravami drugih podjetij. Nestova naprava za zaznavanje dima lahko komunicira preko aplikacije podjetja GE ali preko *Home Connect* z električnimi pečicami in jim

v primeru zaznanega dima pošlje ukaz, naj se ugasnejo. Nestov termostat, ki skrbi tudi za manjšo porabo energije, lahko komunicira z Whirlpoolovimi in Jenn-Air gospodinjskimi aparati. Termostat ima možnost, da zazna, kdaj smo doma in kdaj smo odsotni. Lahko nastavimo, da se pomivalni stroj zažene šele takrat, ko odidemo od doma, če pa smo pozabili prižgano pečico, nas opozori. Nestov termostat tudi poskrbi, da se začetek cikla pralnega ali sušilnega stroja prestavi na energijsko manj obremenjujoče ure. Lahko se poveže z Amazon Alexo ali Google Assistantom in ga tako upravljamo z govornimi ukazi [7].

2.5 Tehnologija

Trg ponuja veliko različnih komunikacijskih tehnologij za povezovanje naprav pri hišni avtomatizaciji. Ker trend razvoja temelji na brezžični komunikaciji, se bomo osredotočili na najbolj razširjene tovrstne tehnologije. Nekatere so že dolgo dobro znane, kot so na primer WiFi, Bluetooth, Z-Wave ali ZigBee, poznamo pa tudi nekaj novejših in uporabnih, na primer Thread [9] [10].

Večina domov že ima brezžične usmerjevalnike, ki delujejo na tehnologiji **WiFi**, zato je za razvijalce uporaba te tehnologije zelo priročna. Trenutno je najbolj razširjen standard 802.11n, ki ponuja visoko pretočnost podatkov (največ 600 Mb/s, po navadi pa 150-200 Mb/s), a je tudi energijsko potraten. Problem lahko nastane tudi takrat, ko imamo v našem stanovanju veliko naprav, ki se morajo boriti za pasovno širino, kar lahko povzroči počasnejši odziv naprav. Komunikacija poteka na frekvencah 2,4 GHz in 5 GHz.

Pomembna komunikacijska tehnologija je **Bluetooth**, predvsem novejša verzija Bluetooth Low-Energy (BLE) oziroma Bluetooth Smart. Uporabna je predvsem za upravljanje luči ali pametnih ključavnic, zaradi kratkega dosega pa ni koristna pri sistemih, kjer je zaželjen stalen nadzor, zato se je proizvajalci pametnih gospodinjskih aparatov ne poslužujejo pogosto. Zagotavlja manjšo pretočnost podatkov (1 Mb/s) kot WiFi, a je energijsko precej bolj varčna. Bluetooth ima frekvenco 2,4 GHz.

Z-Wave je tehnologija, katere glavni namen je uporaba pri avtomatizaciji doma. Poteka na frekvenci 900 MHz. Nižja frekvenca (večina ostalih brezžičnih produktov deluje na 2,4 GHZ) pomeni, da signal ni tako občutljiv na razne ovire (na primer zidove) in motnje, doseg pa je daljši. Vse naprave s to tehnologijo lahko komunicirajo med sabo, ne glede na tip, verzijo ali znamko. Poraba energije je nizka, prenaša pa lahko majhne pakete podatkov s hitrostjo 100 kb/s. Z-Wave je med drugim zelo uporaben pri krmilnikih luči in senzorjih.

Podobno kot Z-Wave je tudi **ZigBee** tehnologija, ki je namenjena izključno hišni avtomatizaciji. Je nizko cenoven, z nizko porabo energije in temelji na brezžičnem komunikacijskem standardu 802.15.4. V večini predelov sveta deluje na frekvenci 2,4 GHz ter na 784 MHz na Kitajskem, 868 MHz v Evropi in na 915 MHz v Ameriki in Avstraliji. Hitrost prenosa podatkov se gliblje med 20 kb/s (na 868 MHz) in 250 kb/s (na 2,4 GHz) [12]. Uporaben je predvsem za kontrolo luči, termostate, alarmne sisteme in podobno.

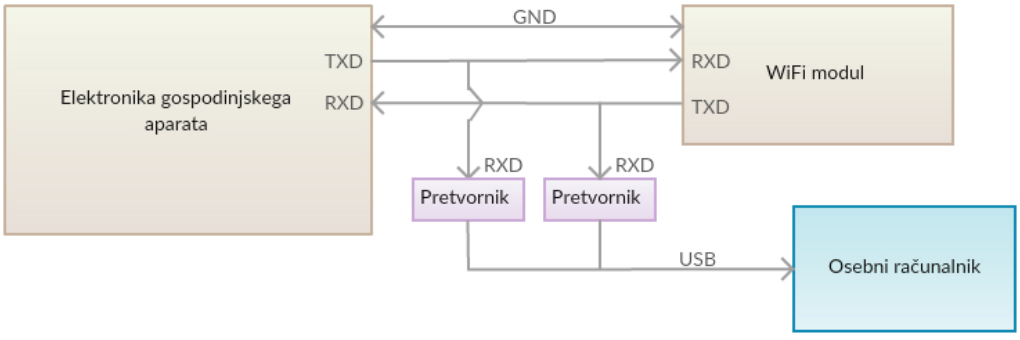
Thread je novejša tehnologija (2014), ki temelji na IPv6 protokolu in 6LowPAN (IPv6 Low-power wireless Personal Area Network). Zagotavlja varno in zanesljivo omrežje z nizko porabo energije. Vanj se lahko poveže preko 250 naprav v domu, lahko pa so povezane tudi v internetne storitve v oblaku. Povežemo lahko gospodinjske aparate, termostate, luči in varnostne sisteme. Podjetje Nest Labs je razvilo odprtokodno implementacijo Threada in ga uporablja za svoje storitve [8]. Tudi Thread teče na frekvenci 2,4 GHz.

Poglavje 3

Delovno okolje

Prvi korak pri povezovanju gospodinjskih aparatov v omrežje je komunikacija med elektroniko gospodinjskega aparata in WiFi modulom, ki sta povezana z RS-232 povezavo. Pri praktičnem delu te diplomske naloge smo na naš računalnik preko USB kabla povezavo RS-232 priključili na WiFi modul. Za povezavo RS-232 z USB kablom potrebujemo dva pretvornika. Celotno povezavo in našo priključitev prikazuje slika 3.1.

Glavni cilj je bil razviti aplikacijo, ki prestreza promet med elektroniko gospodinjskega aparata in WiFi modulom. Pakete, ki jih zajame, nato iz bitov prevede v človeku prijazno oziroma razumljivo obliko, uporabnik pa jih lahko izpiše na zaslonu in tako nadzoruje ta del komunikacije.



Slika 3.1: Prikaz povezave in priključitev z računalnikom.

3.1 WiFi modul

WiFi modul je čip, ki napravi omogoča brezžično komunikacijo z drugo napravo. V radijski signal pretvarja podatke, ki jih dobi od naprave po žici, ter jih oddaja preko antene. Ta tudi lovi radijske valove. WiFi modul ima implementirane potrebne 802.11 protokole, kar omogoča, da se prejeti radijski valovi prevedejo v ustrezno obliko in po žici pošljejo elektroniki naprave.

Pri naši obravnavani komunikaciji se uporablja Panasonicov WiFi modul PAN9320 (slika 3.2). Oddaja in sprejema na frekvenci 2,4 GHz. Sestavljen je iz brezžičnega radia in mikrokontrolerja za integracijo WiFi povezanosti v različnih elektronskih napravah. Ima integriran TCP/IP omrežni sklad s protokoli IPv4, ARP in AutoIP. Vključuje spletni strežnik z AJAX/JSON za spletne aplikacije. Na njem sta dva UART (*Universal Asynchronous Receiver/Transmitter*) vmesnika za nadzor in pregled podatkov. Konfiguracija IP naslovov je v celoti avtomatska, dostop do naprav pa je mogoč po imenu (`http://imeNaprave`). Modul podpira TLS/SSL, https in WiFi zaščito (WPA2) za varno povezavo in pretok podatkov. Več podrobnosti o uporabljenem WiFi modulu je dostopnih na povezavi [14].

Slika 3.2: WiFi modul.

3.2 Standard RS-232

RS-232 je standard za serijsko komunikacijo prenosa podatkov. Uporablja se pri serijskih vratih (ang. *Serial Port*), skozi katera poteka prenos podatkov bit za bitom. Originalno je bil razvit za komunikacijo teleprinterjev preko telefonskih linij. Teleprinter je bil preko povezave RS-232 vezan na modem, ta pa je po telefonski liniji klical drug modem [23].

RS-232 definira signale v komunikaciji med dvema tipoma naprav, in sicer med DTE in DCE napravo. DTE (*Data Terminal Equipment*) naprave pretvorijo uporabnikove informacije v signale in obratno. V našem primeru sta takšni napravi računalnik in elektronika gospodinjskega aprata. DCE (*Data Circuit-terminating Equipment* ali *Data Communication Equipment*) so naprave, ki so vmesni člen med DTE napravami in krogom prenosa podatkov. V komunikaciji, ki jo obravnavamo, je takšna naprava WiFi modul. RS-232 vrata so asinhrona in delujejo brez urinega signala.

Najosnovnejša povezava RS-232 potrebuje le tri priključke oziroma signale:

- za pošiljanje podatkov - TXD
- za sprejem podatkov - RXD
- skupna masa - GND

Po TXD (*Transmit Data*) priključku se podatke pošilja, torej ta predstavlja njihov izvor. To so serijski podatki, ki zapuščajo serijska vrata, generira pa jih vezje UART. Po RXD (*Receive Data*) liniji se podatki prejemajo, gre za njihov ponor. Tok bitov gre preko te linije na vhod UARTa v serijskih vratih. Vsi signali teh vrat potrebujejo povratno pot. To je skupna (signalna) masa, GND (*Common Ground*). Ta signal je za serijsko komunikacijo nujno potreben.

Pogosto se uporabljajo tudi signali za strojni nadzor pretoka (ang. *hardware flow-control*).

Za serijsko komunikacijo je potrebnih veliko nastavitev, kot so nastavitev hitrosti, število podatkovnih bitov na znak, pariteta in število končnih bitov na znak. Pri modernih serijskih vratih, ki uporabljajo UART integrirano vezje, se ponavadi večina teh nastavitev nastavi na višjem nivoju, v programski opremi [13].

Hitrost (ang. *Speed* oziroma *Baud Rate*) prenosa podatkov nam pove, koliko bitov na sekundo se pretaka po serijski komunikaciji. Hitrost vrat in hitrost naprave se morata ujemati. Vse hitrosti prenosa podatkov ne delujejo na vseh serijskih vratih. Zajema tudi bite okvirja (končne bite, paritetnega...), zato je dejansko prenos podatkov (*Data Rate*) nekoliko manjši. Največkrat so podprte hitrosti 75, 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 in 115200 bit/s. V našem primeru poteka komunikacija med elektroniko aparata in WiFi modulom s hitrostjo 19200 bit/s.

Število **podatkovnih bitov** (ang. *Data Bits*) na znak je lahko 5, 6, 7 (ASCII), 8 (največkrat) ali 9. Večinoma se pri serijski komunikaciji znotraj bajta biti nanizajo od najmanj pomembnega do najpomembnejšega (*LSB - Least Significant Bit*), redko pa tudi obratno (*MSB - Most Significant Bit*). Ta vrstni red bitov je odvisen od vsakega sistema in praktično ni nikoli del nastavitev vmesnika serijskih vrat, ampak del programskih nastavitev prejemnika in pošiljatelja. Pri obravnavani komunikaciji je podatkovnih bitov 8, kateri se pošiljajo po sistemu LSB.

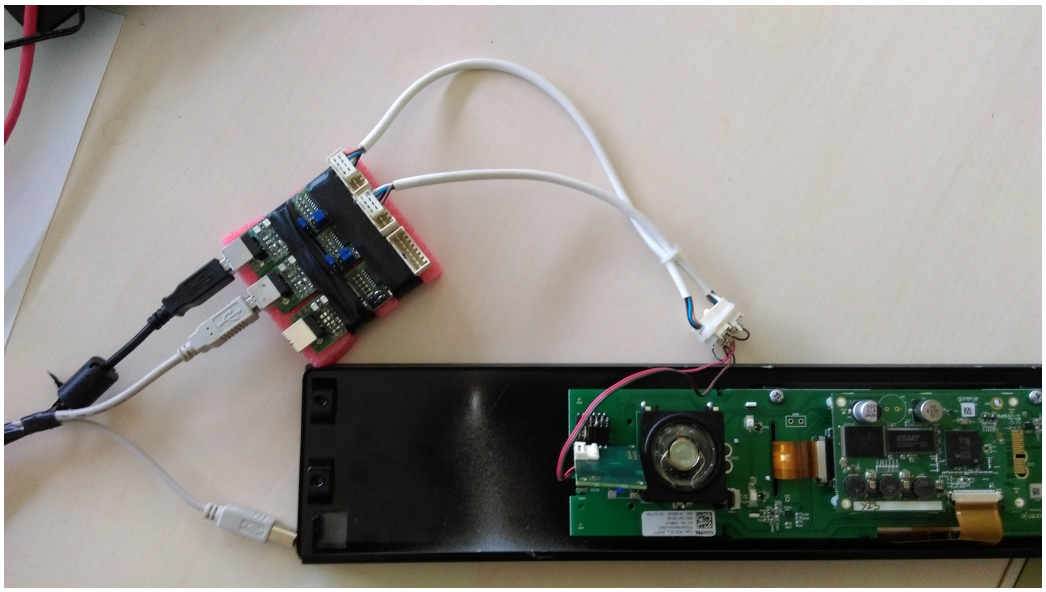
Pariteta (ang. *Parity*) je metoda za zaznavanje napak v prenosu podatkov. Ko je uporabljena pri serijskih vratih, se pri vsakem znaku na koncu doda dodaten bit, tako da je število enic v bitnem zapisu sodo ali liho, odvisno od nastavitev. Če je torej nastavljena na liho (oziroma sodo), število enic pa je sodo (oziroma liho), je zaznana napaka. Težava je, ker ne zazna sodega števila napak. Pariteta je lahko nastavljena na sodo (E - *even*), liho (O - *odd*), na ena (M - *mark*), nič (S - *space*) ali pa, da ni paritetnega bita (N - *none*). Pri obravnavani komunikaciji med elektroniko gospodinjskega aparata in WiFi modulom se uporablja slednja nastavitev.

Končni biti (ang. *Stop Bits*) na koncu bitnega zapisa vsakega znaka

omogočajo napravi, ki prejema signal, da zazna, kdaj se zapis za ta znak konča in da se ponovno sinhronizira s tokom znakov. Elektronske naprave ponavadi uporabljajo en končen bit, in tako je tudi v našem primeru.

Včasih lahko pošiljatelj pošilja hitreje kot lahko sprejemnik procesira. Za ta namen serijske linije pogosto vključujejo **rokovanje** (ang. *Handshaking*), ki se ponavadi razlikuje na strojnem in programskem nivoju. Nastavimo lahko "ukaz za pošiljanje"(RTS - *Request-to-Send*) in "prosto za pošiljanje"(CTS - *Clear-to-Send*), ki omogočata strojni nadzor pretoka podatkov. RTS sporoča, da so podatki na voljo za pošiljanje, CTS pa kdaj je naprava na voljo za prejemanje podatkov. Na programskem nivoju se uporablja XON/XOFF nadzor toka podatkov. XOFF je poslan, kadar želimo, da se pošiljanje podatkov preneha, XON pa, kadar želimo, da se to nadaljuje. Uporabljena sta lahko tudi oba rokovanja hkrati. V našem primeru komunikacije se rokovanje ne uporablja.

Na sliki 3.3 vidimo povezavo dveh USB kablov preko pretvornikov na RS-232 povezavo.



Slika 3.3: Povezava na RS-232 preko USB pretvornika.

3.3 Elektronika gospodinjskega aparata

Elektronika oziroma tiskano vezje mehansko podpira in električno poveže elektronske komponente. Komponente, kot so kondenzatorji, upori ali na primer aktivne komponente, so spajkane na to vezje.

Za nas je pomembna elektronika pri gospodinsjkih aparatih (slika 3.4), ki se praviloma nahaja za nekim grafičnim vmesnikom, na primer zaslonom in/ali gumbi. Skrbi, da se ukazi, ki jih da uporabnik preko grafičnega vmesnika, pošljejo proti podsistemom aparata. Na primer, pečici na zaslonu nastavimo temperaturo, elektronika to zazna, pretvori v bitni zapis in pošlje ukaz proti grelcu.

Slika 3.4: Elektronika gospodinjskega aprata.

Pri našem delu poslušamo komunikacijo z elektroniko pečice podjetja Asko, ki je premijska blagovna znamka Gorenja. Ta pečica ima zaslon na dotik in 4 gumbe (slika 3.5), ki so spojeni na tiskano vezje. Zaslon poganja procesor Toshiba. Na elektroniko je vezan tudi WiFi modul, kateremu se pošiljajo vsi ukazi, ki so zaznani na zaslonu oziroma gumbih, lahko pa preko njega prihajajo tudi ukazi za pečico iz omrežja.

Slika 3.5: Zaslon pečice z gumbi.

3.4 Obravnavana protokola

V obravnavanem delu komunikacije smo naleteli na dva protokola, ki ju je razvilo podjetje Gorenje. Inspector je komunikacijski protokol za pošiljanje paketov med elektroniko in WiFi modulom, medtem ko se XMC uporablja pri komunikaciji elektronike aparata s storitvami v oblaku. XMC podatki se na obravnavani poti, torej med elektroniko aparata in WiFi modulom, prenašajo znotraj Inspector paketov.

3.4.1 Inspector

Inspector je komunikacijski protokol, ki se uporablja za komunikacijo med elektroniko gospodinsjkega aparata in WiFi modulom. Njegov glavni namen je omogočanje proizvajalcu modula preverjanje vseh pomembnih funkcij že med procesom proizvodnje ter zagotavljanje komunikacije gospodinjskega aparata z ostalimi napravami. WiFi modul lahko po protokolu Inspector prejema konfiguracijske parametre s strani gospodinjskega aparata.

Protokol Inspector mora biti implementiran na obeh vpletenih straneh - na elektroniki gospodinjskega aparata in na WiFi modulu.

Da bi bila komunikacija čim preprostejša, se za pošiljanje paketov uporablja standard RS-232. Tip paketa je odvisen od poslanih podatkov, zato dolžina paketa ni fiksna.

Slika 3.6: Zgradba paketa Inspector protokola.

Slika 3.6 prikazuje zgradbo paketa protokola Inspector. Kot vidimo, je sestavljen iz naslednjih delov:

- SOF
  - označuje začetek paketa (ang. *Start Of Frame*)
  - šestnajstiška vrednost bajta: FD
  - vrednost enaka pri vseh paketih
- Tip paketa (*Frame Type*)
  - določa, katera vrsta podatkov se prenaša v paketu
  - možni tipi:
    - Hello - paket poslan ob vzpostavitvi komunikacije
    - Empty - paket za vzdrževanje komunikacije
    - Error - paket poslan v primeru napake v komunikaciji
    - ACK - potrditev prejetega paketa
    - NACK - odgovor na neznano sporočilo
    - App/Data - paket z ostalimi podatki
- Številka paketa (*Frame Number*)

  - določa postopno vrednost (ang. *incremental value*) vsakega poslanega paketa
  - uporablja se za potrditev prejetih paketov

- Podatki (ali tovor - *Payload*)
  - informacije, ki se prenašajo
  - lahko vsebuje XMC podatke

- CRC
  - kontrolna vsota (ang. *Cyclic Redundancy Check*)
  - za odkrivanje napak
  - negativna vsota tipa paketa, številke paketa in podatkov

- EOF
  - označuje konec paketa (ang. *End Of Frame*)
  - šestnajstiška vrednost bajta: FE
  - vrednost enaka pri vseh paketih

Za pravilno vzpostavljeno komunikacijo med dvema enotama (elektroniko naprave in WiFi modulom) morajo biti komunikacijski parametri nastavljeni na:

- hitrost: 19200
- pariteta: off
- končni biti: 1

3.4.2 XMC

XMC je protokol, ki se uporablja za komunikacijo med elektroniko aparata in stroritvami v oblaku (slika 3.7). Po njem se prenašajo ukazi gospodinjskemu aparatu s strani uporabnika ter informacije, kaj se s tem aparatom dogaja. Če želimo torej na pečici nastaviti peko z določenimi nastavitvami (čas, temperatura, način...), se ti podatki po XMC protokolu pošljejo na WiFi modul in od tu naprej v omrežje.

Slika 3.7: Diagram komunikacije.

Izpeljan je iz označevalnega jezika XML. Prevzema njegovo strukturo, ne pa tudi določenih lastnosti, kot so atributi ter opisna imena značk. Značke in skoraj vse informacije so predstavljene z numeričnimi vrednostmi.

Med komunikacijo in pošiljanjem naprava vedno doda na konec XMC podatkov znak '#' (lojtra), medetm ko strežnik doda znak '.' (pika).

Primera:

1. XMC podatke pošilja naprava

```
<0>
    <2>
        <1999>
            <2010>2011</2010>
            <2030>2031</2030>
            <2050>2051</2050>
        </1999>
    </2>
</0>#
```

2. XMC podatke pošilja strežnik

```
<0>
    <2>
        <1999>
            <2010>∼</2010>
            <2030>∼</2030>
            <2050>∼</2050>
        </1999>
    </2>
</0>.
```

***Opomba:** Znak '∼' v tem primeru predstavlja branje parametra naprave. Potem, ko strežnik pošlje takšno zahtevo, naprava odgovori s podobnim sporočilom kot ga predstavlja primer v 1. točki.*

Struktura XMC podatkov se začne s korensko značko '<0>'. Ta lahko vsebuje eno ali več značk, znotraj katerih se prenašajo specifični podatki. Nekatere izmed pomembnejših značk na drugem nivoju:

- <1> - za predstavitev akcij naprave
- <2> - za predstavitev stanj naprave
- <3> - za odpravljanje napak
- <5> - odgovor naprave/strežnika, kadar zazna napako v prijetih XMC podatkih
- <9000> - za podatke o proizvajalcu in lastnostih (tip, model) naprave

Vsaka povezana naprava pošlje statusni "keep-alive"paket vsakih 5 sekund. Glede na ta statustni paket je lahko vsaka naprava prepoznana po njenem tipu. Naprava začne pošiljati te pakete po tem, ko prejme začetni "hello"paket in je povezava vzpostavljena. Ti paketi so nekoliko drugačni, saj nimajo korenske značke '<0>', na koncu pa nimajo lojtre. Poslana je

zgolj številka naprave znotraj značke '<4>'.

Primer statusnega paketa:

```
<4>2042</4>
```

Naprava, ki ima implementiran XMC, prepozna prejeti ukaz glede na značke na drugem, tretjem in včasih četrtem nivoju. Na zadnjem nivoju se v značkah nahajajo parametri, ki jih naprava prejme.

Za nas bodo najpomembnejši podatki, ki imajo na drugem nivoju značko '<5>', torej je naprava (ali strežnik) odgovorila, da je prejela podatke z napako. Do sedaj so znane tri možne napake v podatkih:

1. Celotno XMC sporočilo je pomanjkljivo ali vsebuje napake.
2. Nekatere vrednosti parametrov so napačne.
3. Nekateri parametri niso v dosegu funkcionalnega profila naprave.

Primer XMC sporočila z napako (1. možnost):

a) Naprava prejme (nepopolno strukturo):

```
<0>
    <2>
        ...podatki...
    </2>
</
```

b) Naprava odgovori:

```
<0>
    <5>
        <0>
            <2>
                ...podatki...
            </2>
        </
    </5>
</0>
```

3.5 Izbira razvojnega okolja in tehnologije

Pred začetkom dela smo se odločili, da bomo aplikacijo razvijali v programskem jeziku C#. Ta jezik ima implementiranih veliko primernih knjižnic za glavne funkcionalnosti naše aplikacije - za poslušanje serijske komunikacije, delo s podatkovno bazo, delo s stringi (nizi znakov), izvoz v Excelov dokument... Za razvojno okolje smo si izbrali Microsoft Visual Studio, saj omogoča pregledno in učinkovito delo s programskim jezikom C#. Razvoja grafičnega uporabniškega vmesnika smo se lotili z WPF (*Windows Presentation Foundation*) grafičnim podsistemom, kjer strukturo definiramo z označevalnim jezikom XAML (*Extensible Application Markup Language*).

Aplikacijo smo razvijali po vzorcu *Model - View - ViewModel* (MVVM), ki je prilagojen vzorec *Model - View - Controller* (MVC) predvsem za Microsoftove aplikacije razvite z označevalnim jezikom XAML. **View** (pogled) je odgovoren za definiranje strukture, razporeditve in prikaza, kar uporabnik vidi na zaslonu. Zaželjeno je, da je pogled definiran zgolj z XAML z zelo malo kode v ozadju, ki ne vključuje poslovne logike. **Model** vključuje podatkovni model s poslovno in validacijsko logiko. **ViewModel** deluje kot vmesni člen med pogledom ter modelom in je odgovoren za upravljanje z logiko pogleda. Tipično ViewModel komunicira z modelom preko metod, ki jih lahko kličemo, znotraj razredov modela. View model pogledu zagotavlja podatke, ki jih dobi iz modela, v takšni obliki, da jih lahko pogled takoj uporabi.

Poglavje 4

Razvoj aplikacije

Ko smo se fizično povezali na WiFi modul, smo lahko začneli razvijati aplikacijo. Z njo lahko prestregamo pakete protokola Inspector. Ti se nato pretvorijo v človeku razumljivo obliko ter se ob želji uporabnika izpišejo na zaslon. Glavni cilj aplikacije je torej nadzor komunikacije med elektroniko gospodinjskega aparata in WiFi modulom.

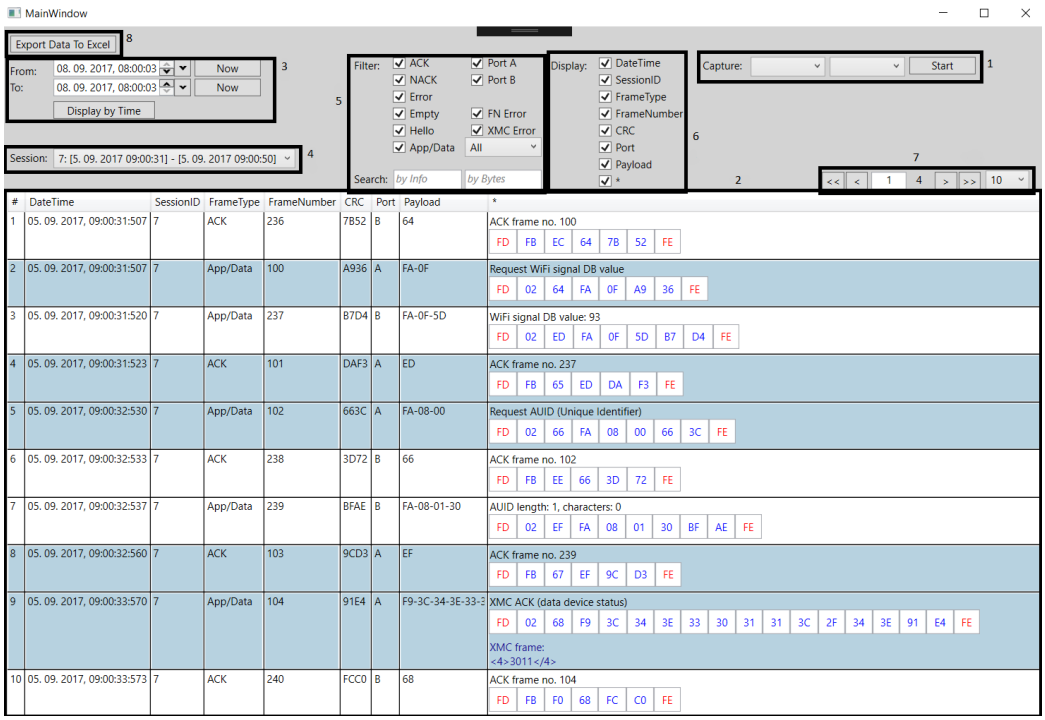
Aplikacija diagnosticira specifične napake, ki se lahko zgodijo med komunikacijo. Zazna, če se pojavi paket, ki ima isto številko okvirja (*Frame Number*) kot njegov predhodnik, nazorno (z rdečim obarvanjem paketa) pa nam prikaže, če je prišlo do napake v podatkih z XMC vsebino ali pa, če je protokol Inspector zaznal napako (tip paketa je Error).

Izgled (*View*) uporabniškega vmesnika smo določili v XAML datoteki `MainView.xaml`. Ker smo aplikacijo razvijali z *Model - View - ViewModel* (MVVM) pristopom, smo logiko za delovanje grafičnega vmesnika (*View Model*) implementirali v razredu, ki smo ga poimenovali `MainViewModel`.

Slika 4.1 prikazuje uporabniški vmesnik aplikacije. Sestavljen je iz osmih komponent, ki predstavljajo glavne funkcionalnosti aplikacije:

1. Zajemanje paketov
2. Polje za izpis paketov
3. Izbira paketov glede na čas

4. Izbira paketov glede na sejo
5. Filtriranje paketov
6. Izbira prikaza stolpcev
7. Ostranjevanje
8. Izvoz paketov v Excelov dokument



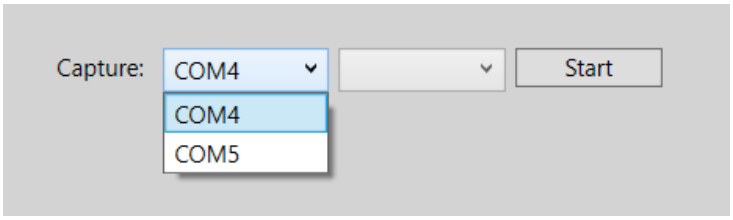
Slika 4.1: Zaslonski prikaz aplikacije.

4.1 Zajemanje paketov

Glavna funkcionalnost aplikacije je prestrezanje komunikacije med WiFI modulom in elektroniko gospodinsjkega aparata. Gre za zajemanje paketov protokola Inspector. Te dobi računalnik na svoj USB vhod.

Za občutek o velikosti problema smo z aplikacijo eno uro prestregali promet med WiFi modulom in elektroniko pečice. V tem času je bilo prestreženih 9164 paketov, a medtem pečice nismo uporabljali. Če bi jo, bi se lahko številka konkretno povečala. To je odvisno od same intenzivnosti uporabe pečice. Za vsako izbiro/spremembo na zaslonu uporabniškega vmesnika pečice se WiFi modulu pošlje paket protokola Inspector, ki vsebuje XMC podatke.

Za začetek prestrezanja uporabnik najprej iz dveh spustnih menijev izbere dvoje vrat, na katerih bo program prestrezal komunikacijo (na sliki 4.2 COM4 in COM5). Prva izbira se imenuje A, druga pa B. Prikaz vrat, ki so na voljo za izbiro, se posodobi ob vsakem kliku na spustni meni. S klikom na gumb "Start" se začne prestrezanje komunikacije.



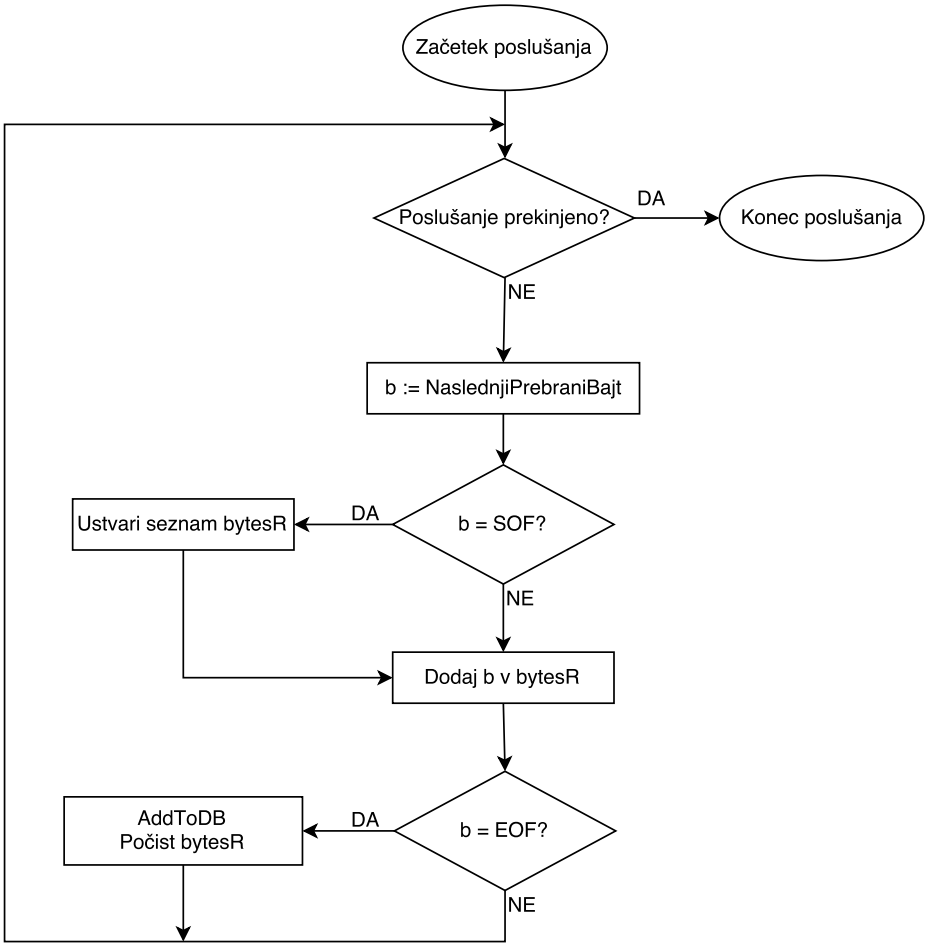
Slika 4.2: Polje za zajemanje paketov.

Za poslušanje komunikacije smo implementirali razred, ki ga imenujemo `Sniffer`. Konstruktor tega razreda dobi parameter `COM`, ki v našem primeru predstavlja ime USB vhoda oziroma vrata. Znotraj razreda ustvarimo objekt razreda `SerialPort` iz imenskega prostora `System.IO.Ports`. Objektu razreda `SerialPort` določimo parametre, ki so potrebni za poslušanje protokola Inspector, z njimi pa podamo tudi parameter `COM`. Programsko odpremo povezavo in v niti (`Thread`) beremo po bajtih, kar dobimo na naša serijska vrata.

Znotraj niti, v kateri beremo, preverjamo, ali je prebrani bajt enak začetku Inspector paketa (SOF - šestnajstiška vrednost FD) oziroma, če je enak končnemu bajtu (EOF - šestnajstiška vrednost FE). Bajte od SOF do EOF (vključno s tema dvema) shranjujemo v seznam bajtov `bytesR`. Ta seznam

torej predstavlja en prestrežen paket protokola Inspector.

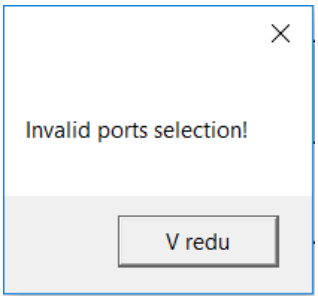
Prestrežen paket želimo ustrezno razbiti, dobiti iz zapisa v bajtih ustrezne podatke ter ga shraniti v lokalno podatkovno bazo. To storimo v funkciji `AddToDB` (dodaj v bazo), ki jo bomo podrobneje opisali v nadaljevanju. Slika 4.3 predstavlja diagram poteka branja bajtov na serijskih vratih.



Slika 4.3: Diagram poteka branja iz serijskih vrat.

Ker poslušamo dvosmerno komunikacijo, moramo ustvariti dva objekta razreda `Sniffer`, vsakemu podamo ime enega izmed dveh vhodov, na katera prihaja prestrežena komunikacija (na primer COM4 in COM5), ter ime izbire vrat glede na spustni meni (A ali B). Ob izbiri vrat preverimo, če je ta

veljavna - to pomeni, da je uporabnik izbral obojna vrata, ter da je izbral dvoje različnih vrat. V kolikor izbira ni veljavna, se uporabniku pojavi okno z obvestilom o napaki, kot kaže slika 4.4.



Slika 4.4: Okno z obvestilom o napaki.

Ko uporabnik začne prestrezati komunikacijo, torej klikne na gumb "Start", se na temu gumbu spremeni napis v "Stop". Ta gumb torej uporabimo tudi za prenehanje poslušanja komunikacije.

4.1.1 "Byte Stuffing"

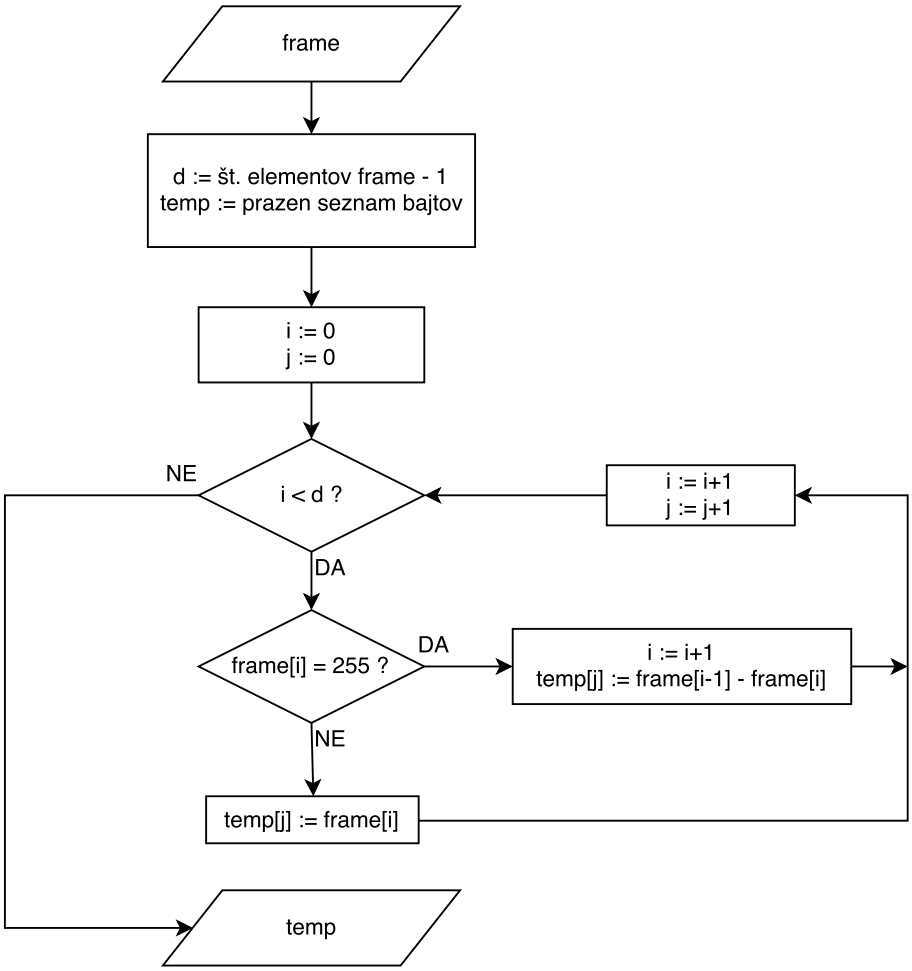
V zapisu paketa protokola Inspector se lahko zgodi, da ima kakšen bajt znotraj paketa šestnajstiško vrednost FD ali FE, torej enako, kot jo ima oznaka za SOF ali EOF. V tem primeru se ta nadomesti z dvema bajtoma po naslednjem ključu:

- FD →FF-02
- FE →FF-01

Bajt z zapisom FF torej označuje "byte stuffing"oziroma *polnjenje bajtov*. Da ne pride do zmede, se tudi bajt FF nadomesti z dvema, in sicer z FF-00. Ker lahko bajte predstavimo številsko (FF kot 255, FD kot 254, FE kot 253, 01 kot 1 in 02 kot 2), je torej zamenjava definirana z odštevanjem.

Primer: FE = FF - 01 →254 = 255 - 1

Na sliki 4.5 vidimo diagram potega funkcije, s katero iz zapisa paketa ustrezno nadomestimo dvojice bajtov, ki so posledica "byte stuffinga", z enim samim bajtom.



Slika 4.5: Diagram poteka funkcije za rešitev "byte stuffinga".

Funkcija kot parameter prejme seznam bajtov `frame`, ki predstavlja zapis paketa protokola Inspector, v obliki, kot ga računalnik prestreže. Ti bajti so torej tipa `byte`, kar pomeni, da so predstavljeni številsko. Znotraj funkcije se sprehodimo po tem seznamu in za vsak bajt preverimo, če je enak 255. Če je, od njega odštejemo njegovega naslednika in to vrednost shranimo v seznam `temp`. Če pa ni, pa bajt samo prepišemo v `temp`. Ta seznam torej

predstavlja zapis paketa v bajtih brez "byte stuffinga".

Primer:

Med prestrezanjem komunikacije smo prestregli paket:

`FD-02-FF-02-FA-08-01-30-FF-00-77-FE`

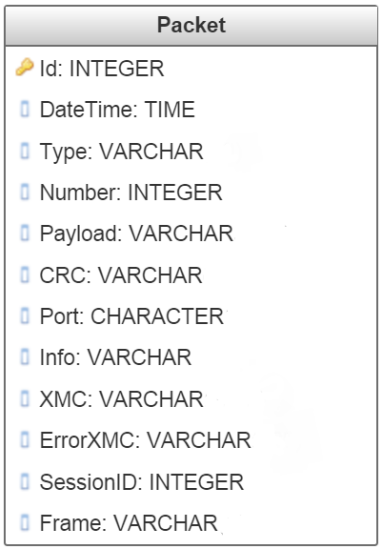
Zapis paketa po odpravi "byte stuffinga":

`FD-02-FD-FA-08-01-30-FF-77-FE`

4.1.2 Vzpostavitev lokalne baze in shranjevanje paketov

Ker želimo, da ima uporabnik možnost stalnega dostopa do prestreženih paketov, te shranjujemo v lokalno podatkovno bazo. Pri našem delu smo uporabili kar bazo, ki jo ponuja Microsoft Visual Studio, torej Service-based Database. Po tej bazi poizvedujemo s SQL (*Structured Query Language*) jezikom.

Sestava podatkovne baze je enostavna, saj vsebuje le eno entiteto, ki smo jo poimenovali *Packet*. Entiteta z atributi je prikazana na sliki 4.6:



Slika 4.6: Sestava podatkovne baze.

Primarni ključ elementov je *Id*, ki se generira sam pri shranjevanju paketa v bazo. *DateTime* predstavlja datum in čas shranjevanja v bazo, *Type* tip paketa (Hello, ACK, NACK, App/Data, Empty ali Error), *Number* pa številko paketa. Atribut *Payload* so podatki (tovor), ki jih prenaša paket. Predstavljeni so v bajtih, v bazo pa jih zapišemo v šestnajstiškem zapisu. Prav tako *CRC*, ki predstavlja kontrolne bite, pod atribut *Port* pa shranimo, na katerih vratih je bil prestrežen paket. *Info* uporabniku pove, katere informacije nosi paket. Pod *XMC* shranimo morebitne XMC podatke, ki se prenašajo, pod *ErrorXMC* pa v primeru napake v XMC podatkih opis te napake. Shranimo tudi številko seje pod *SessionID*. *Frame* predstavlja zapis celotnega paketa, kar nam bo kasneje prišlo prav. Pod atribut *Frame* shranjujemo direktno prestrežen paket, medtem ko pred shranjevanjem ostalih lastnosti paketa odstranimo "byte stuffing".

Za implementacijo shranjevanja paketov v bazo ustvarimo novo funkcijo `AddToDB` v razredu `Sniffer`, ki kot parameter prejme seznam prestreženih bajtov. Znotraj funkcije uporabimo razrede iz imenskega prostora `System.Data.SqlClient`. Pakete shranjujemo s kreiranjem SQL ukaza, ki je oblike:

```
INSERT into Packet ([DateTime], ... , [Frame])
     VALUES (@DateTime, ... , @Frame)
```

Temu moramo podati parametre (na primer `@DateTime`). *Payload*, *CRC* in *Frame* pridobimo direktno iz prestreženega zapisa v bajtih, potrebna je le konverzija v tip `string`, *Number* pa shranimo v obliki številke, ki jo predstavlja ustrezen bajt. V *DateTime* shranimo `System.DateTime.Now`, ki nam priskrbi trenutni čas in datum do tisočinke natančno. Ta torej predstavlja čas shranjevanja paketa v bazo. Časovna natančnost nam pride prav, saj je paketov v komunikaciji med WiFi modulom in elektroniko gospodinjskega aparata lahko v kratkem času zelo veliko. *Type* dobimo glede na drugi bajt v zapisu. Ta nosi podatek, za kateri tip gre.

Če zaznamo, da gre za tip "App/Data", potem preverimo, če vsebuje

XMC podatke. Če jih, potem bajte *Payload* dela paketa pretvorimo v ASCII zapis. Ta zapis shranimo v bazo pod *XMC*, torej teh podatkov nadalje ne obdelamo. Preverimo še, če so XMC podatki slučajno odgovor na napako. Če so, potem pod *ErrorXMC* zapišemo podatke o napaki.

Ko želimo shraniti, iz katere strani prihaja določen paket (*Port*), uporabimo parameter z imenom izbire vrat (A ali B). Dve različni oznaki želimo imeti predvsem zato, ker pakete, ki prihajajo iz ene strani, zaradi preglednosti v uporabniškem vmesniku obarvamo drugače kot tiste, ki prihajajo iz druge strani.

Za pridobitev informacij o paketu (*Info*) pregledamo vse možne informacije, ki jih lahko daje paket. Teh je veliko in niso tema te diplomske naloge, zato jih podrobno ne bomo predstavili. Primer zapisa informacij: "*WiFi is connected with router*"("WiFi je povezan z usmerjevalnikom").

V nadaljevanju razvoja aplikacije bomo nekako morali ločevati, kateri paketi so bili prestreženi v isti seji. Temu podatku rečemo *SessionID* in ga dodamo šele ob koncu aktualnega prestrezanja. Vpeljemo novo fukncijo, ki se kliče ob koncu vsakega prestrezanja. Znotraj nje se najprej povežemo na podatkovno bazo in pridobimo zadnji zadetek v bazi oziroma zgolj zadnji *SessionID*. V bazi nato posodobimo vse vnose paketov (SQL stavek UPDATE), ki so bili zajeti v tej seji, in jim nastavimo za ena povečan pridobljen *SessionID*. Če v prejšnji poizvedbi ni zadetka, nastavimo *SessionID* na ena. Vnose, ki so bili dodani v aktualni seji, dobimo glede na čas začetka in konca prestrezanja, ki si ju shranimo.

4.2 Polje za izpis paketov

Prostor, ki je namenjen izpisu razčlenjenih paketov protokola Inspector, zavzame v uporabniškemu vmesniku največ prostora. Izpisujemo pakete z lastnostmi, ki jih postavimo v 9 stolpcev:

1. #
2. DateTime
3. SessionID
4. FrameType
5. FrameNumber
6. CRC
7. Port
8. Payload
9. *

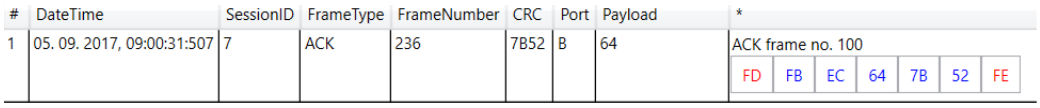
V stolpec DateTime izpisujemo datum in čas shranjevanja paketa v bazo, torej ta podatek pridobimo iz podatkovne baze, podobno kot tudi SessionID (ID seje), FrameType (tip paketa - v bazi *Type*), FrameNumber (številka paketa - v bazi *Number*), CRC (kontrolni biti - izpisujemo v šestnajstiškem zapisu), Port (oznaka vrat prestrezanja - A ali B) in Payload (tovor paketa). Pod # izpisujemo številko, pod katero je bil paket pridobljen iz baze. Prvi zadetek ima številko 1, drugi 2 in tako dalje.

Pod * izpisujemo več podatkov: zapis informacije o paketu (*Info*), zapis celotnega paketa v bajtih, morebitno XMC sporočilo ter morebitno poročilo o napaki v XMC podatkih (*ErrorXMC*). Vse te podatke pridobimo iz baze.

Celoten paket v bajtih izpišemo v obliki seznama bajtov (v šestnajstiškem zapisu). Aplikacija omogoča, da uporabnik s klikom izbere določen bajt. Ta

opcija je implementirana za bodoči razvoj aplikacije, ko bo uporabnik morda želel še preglednejši pomen vsakega bajta.

Pri poslušanju komunikacije smo 5. 9. 2017 ob 9:00 prestregli paket na vratih z oznako B. Šestnajstiški zapis bajtov tega paketa je `FD-FB-EC-64-7B-52-FE`. Slika 4.7 prikazuje izpis tega paketa na zaslonu.



Slika 4.7: Pogled na izpisan paket.

V programski kodi smo za potrebe lažjega izpisa paketov, filtriranja, obarvanja in zaznavanja napak implementirali razred `FrameModel`. Ta ima atribute, ki pripadajo atributom entitete *Packet* iz baze: `ID`, `DateTime`, `Type`, `Number`, `Payload`, `CRC`, `Port`, `Info`, `XMC`, `ErrorXMC`, `SessionID` in `All` (predstavlja *Frame* iz baze). Poleg teh ima še dva atributa za potrebe obarvanja paketov (`Ack`, `IsRelated`), `Line` (kateri zaporedni zadetek je - # stolpec v izpisu), `Page` (za potrebe ostranjevanja) ter `FrameNumberError`, ki označuje, če je bila pri paketu zaznana napaka številke paketa.

Ko pridobimo pakete iz baze, se sprehodimo skozi njih in vsakemu paketu kreiramo pripadajoč objekt razreda `FrameModel`. Na istem mestu tudi preverjamo, če se je pri kakšnem paketu pojavila napaka številke paketa. To storimo tako, da si v zanki za vsako oznako vrat (A in B) beležimo prejšnji obiskani paket oziroma objekt razreda `FrameModel`. Če ima trenutno obravnavani paket enako številko paketa kot njegov predhodnik na istih vratih, pri njem označimo, da je bila zaznana napaka številke paketa.

4.2.1 Obarvanje paketov

Za večjo preglednost smo dodali obarvanje paketov oziroma njihovih delov. V osnovnem izpisu so tisti paketi, ki so bili prestreženi na vratih (*Port*) z

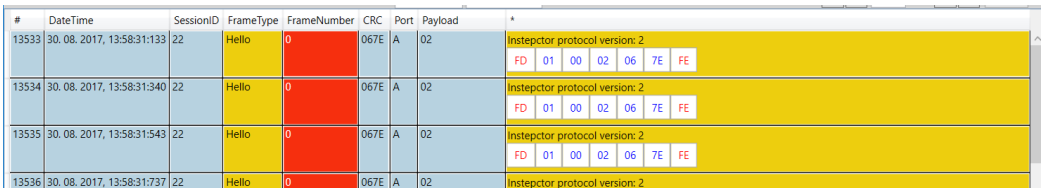
oznako A, drugače obarvani kot tisti, ki so iz vrat B. Pakete iz A obarvamo modro, iz B pa belo. To uporabniku omogoča jasno razločevanje med paketi glede na stran izvora.

Večina prestreženih paketov je tipa ACK ali tipa App/Data. Ker želimo jasno ločiti pakete tipov Hello, Error in NACK, se njihova polja v stolpcih * in FrameType obarvjo po naslednjem ključu:

- tip paketa Hello - rumeno
- tip paketa Error - svetlo rdeče
- tip paketa NACK - oranžno

V aplikaciji zaznavamo dve specifični napaki - *XMC Error* (odogovor na napako v XMC podatkih) in *FrameNumber Error* (napaka številke paketa). Če se pojavi prva, se celica v stolpcu * pri paketu, ki jo je zaznal, obarva temno rdeče. Drugače obarvamo tudi pakete, pri katerih je bila zaznana napaka številke paketa. Pri teh paketih se celica v stolpcu FrameNumber obarva rdeče.

Pri prestrezanju komunikacije smo naleteli na primer, ko Hello paket ni dobil odgovora, zato je bil večkrat poslan. Naša aplikacija je v tem primeru zaznala napako številke paketa. Slika 4.8 prikazuje odziv aplikacije.



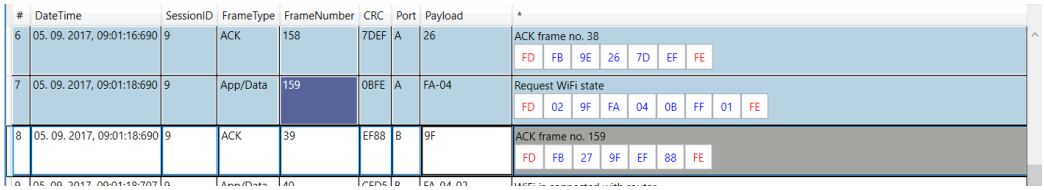
Slika 4.8: Napaka številke paketa.

Ko uporabnik klikne na določen paket, se njegova celica v stolpcu * obarva sivo, cel paket pa dobi črno obrobo. Tako uporabnik jasno loči, kateri paket je izbral.

Če je izbrani paket tipa ACK, se številka paketa (celica v stolpcu FrameNumber), ki ga potrjuje, obarva vijolično, kar uporabniku omogoča pregledno

zaznavanje, kateri paket določena ACK pošiljka potrjuje. Pri implementaciji smo si pomagali z atributom `Ack` razreda `FrameModel`. V primeru, da je nek paket tisti, ki ga ACK potrjuje, mu ta atribut nastavimo na `true`. To storimo tako, da pogledamo, katero številko ACK potrjuje, in iščemo paket s to številko. Teh je lahko več, zato izberemo tistega, ki je glede na ID najbližje.

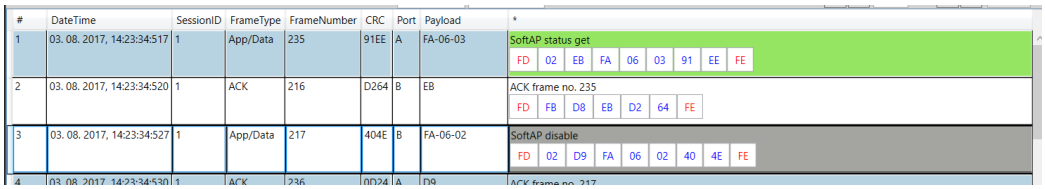
Slika 4.9 obarvanje potrjenega paketa. S klikom smo izbrali paket pod zaporedno številko (#) 8, ki je tipa ACK. Ta potrjuje paket pod zaporedno številko 7.



Slika 4.9: Obarvanje potrjenega paketa.

Če je izbrani paket (na katerega uporabnik klikne) tipa App/Data, potem se celice v stolpcu * pri paketih s sorodno vsebino obarvajo zeleno. Paketi s sorodno vsebino obravnavajo enak problem. Na primer, če uporabnik klikne na paket, s katerim želi gospodinjski aparat izvedeti moč WiFi signala, se zeleno obarvajo paketi, ki obravnavajo moč WiFi signala.

Na sliki 4.10 smo s klikom izbrali paket v tretji vrstici, zato je celica v stolpcu * obarvana sivo. Paket v prvi vrstici mu je soroden, v tem primeru je izbrani paket odgovor na prvega, zato je celica obarvana zeleno.



Slika 4.10: Obarvanje sorodnega paketa.

Včasih se zgodi, da določenemu delu paketa pripada dvojno obarvanje. Takrat moramo določiti vrstni red oziroma hierarhijo barvanja. Na primer, če je s klikom izbrani paket odgovor na XMC napako, moramo določiti, ali se bo celica v stolpcu * obarvala rdeče ali sivo. Zato smo uvedli naslednjo hierarhijo:

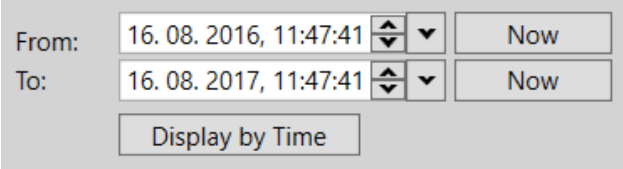
1. izbira paketa (siva celica v stolpcu *)
2. XMC Error (temno rdeča v *) in FN Error (rdeča v FrameNumber)
3. tip paketa - Hello (rumena v * in FrameType), Error (svetlo rdeča), NACK (oranžna)
4. sorodni paketi (zelena v *) in potrjeni paket z ACK (vijolična v FrameNumber)
5. obarvanje glede na oznako vrat (moder/bel celoten paket)

To pomeni, da je barvanje izbranega paketa najpomembnejše, glede na oznako vrat pa najmanj pomembno. V primeru s klikom izbranega paketa, ki je odgovor na XMC napako, bo torej celica v stolpcu * obarvana sivo.

4.3 Izbira paketov glede na čas

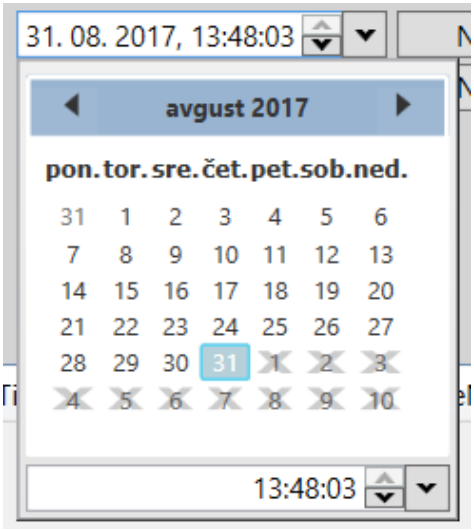
Aplikacija omogoča uporabniku, da izbere pakete, ki jih želi izpisati, glede na čas prestrezanja. Uporabnik v polju, ki ga prikazuje slika 4.11, izbere začetni in končni čas prestrezanja, ob kliku na gum "Display by Time", pa se na zaslon izpišejo vsi paketi, ki so bili zajeti znotraj tega časovnega intervala. V tem primeru aplikacija dostopa do baze in pridobi podatke glede na atribut DateTime.

Standardni Microsoft Visual Studio znotraj svojega nabora WPF kontrol ne pozna elementa, pri katerem bi enostavno izbrali čas in datum, zato smo si pri našem delu pomagali z Xceedovim paketom WPF kontrol "Extended WPF Toolkit". Ta vsebuje element `<DateTimePicker>`, ki smo ga v tem



Slika 4.11: Polje za prikaz paketov glede na čas.

primeru uporabili. Ta omogoča, da izberemo datum in čas, kot prikazuje slika 4.12. Datum lahko uporabnik izbere z menijem, ki prikazuje koledar, lahko pa ga tudi prosto vpiše, kot to stori z željeno uro.



Slika 4.12: DateTimePicker: izbira datuma in čas.

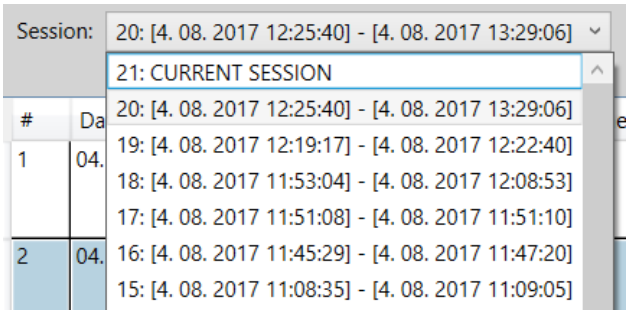
Pri implementaciji smo poskrbeli, da željeni konec prestrezanja ne more biti manjši od željenega začetka in obratno. To smo storili z omejitvijo glede na vnos v preostalem polju.

Vsakemu izboru datuma in časa smo dodali še gumb "Now". Ob kliku nanj se povezana vrednost nastavi na trenutni čas. To uporabniku pride prav, kadar želi izpisati pakete med aktualnim prestrezanjem. V primerjavi z izpisom glede na sejo lahko v tem primeru tudi zaključi sejo (aktualno prestrezanje) in nato začne še eno na novo, paketi pa se mu bodo izpisali v enem

kosu. V primeru klika na gumb "Now" se pri začetnem času prestrezanja ta vseeno nastavi na trenutnega, čeprav seveda ni manjši od končnega.

4.4 Izbira paketov glede na sejo prestrezanja

Poleg izpisa po času prestrezanja ima uporabnik tudi možnost, da izpiše pakete glede na sejo. To stori tako, da v spustnem meniju (slika 4.13) izbere ustrezno sejo. Na zaslon se nato izpišejo paketi, ki so urejeni glede na njihov čas shranjevanja v bazo (prvi izpisan paket je bil prvi v bazi).



Slika 4.13: Polje za prikaz paketov glede na sejo.

Seznam sej napolnemo s funkcijo, ki vrača seznam nizov oblike:

```
SessionID: [čas prvega paketa v seji] - [čas zadnjega paketa v seji]
```

V funkciji se povežemo na bazo, od koder pridobimo vse številke sej (*SessionID*), nato pa še čas (*DateTime*) prvega in zadnjega prestreženea paketa v vsaki posamezni seji. Te podatke dodamo v niz, ki se shrani v seznam. Dodamo pa tudi opcijo za prikaz poslušanja "v živo", torej paketov, ki jih ravno prestrežemo. Za to izberemo niz oblike:

```
SessionID: CURRENT SESSION
```

V tem primeru dobimo *SessionID* tako, da za ena povečamo zadnji *Ses-*

sionID, ki smo ga dobili iz baze. Če uporabnik izbere to možnost, se mu vsako sekundo posodobi izpis paketov na zaslonu. V tem primeru so paketi razvrščeni po času padajoče - prvi paket je bil nazadnje shranjen v bazo.

Seznam sej se posodobi ob vsakem odprtju spustnega menija. S tem zagotovimo, da je ob aktivnem prestrezanju paketov na voljo prestrezanje "v živo", ter da se ob koncu prestrezanja (konec neke seje) končana seja ob odprtju spustnega menija doda v seznam.

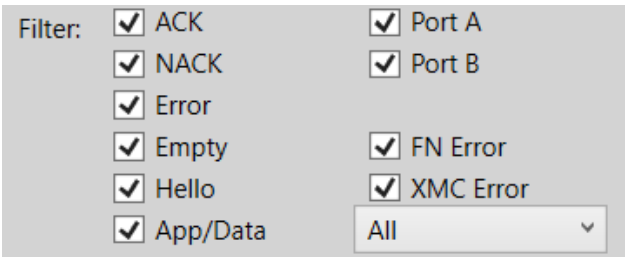
4.5 Filtriranje paketov

Ker želimo uporabniku omogočiti čimbolj pregleden in uporaben izpis paketov, smo aplikaciji dodali možnost filtriranja rezultatov. Uporabnik lahko izpis paketov filtrira glede na:

- tip paketa (ACK, NACK, Error, Empty, Hello, App/Data)
- oznako vrat (Port A, Port B)
- specifično napako (FN Error, XMC Error)

Pakete filtriramo glede na lastnosti objektov razreda `FrameModel`, ki so shranjeni v seznam za izpis na zaslon.

Te filtre uporabnik vklaplja/izklaplja z obkljukanjem kvadratka pred željenim filtrom. Na sliki 4.14 ni vključen noben filter (vsi kvadratki so obkljukani).

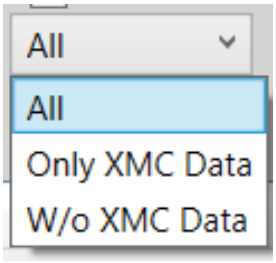


Slika 4.14: Filtriranje paketov.

V kolikor uporabnik ne filtrira po tipu App/Data, torej se mu paketi tega tipa izpišejo, lahko izbere enega izmed treh podfiltrov tega tipa:

- *All* (izpis vseh paketov - z XMC vsebino ter brez nje)
- *Only XMC Data* (izpis le tistih paketov, ki vsebujejo XMC podatke)
- *W/o XMC Data* (izpis le tistih paketov, ki ne vsebujejo XMC podatkov)

Uporabnik ta podfilter izbere v spustnem meniju, ki je prikazan na sliki 4.15. Če filtrira po tipu App/Data, potem je ta spustni meni zaklenjen in ga ne more odpreti.

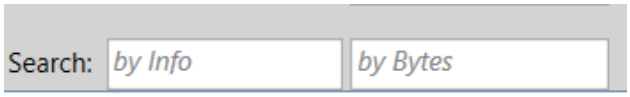


Slika 4.15: Filter paketov glede na XMC podatke.

Poseben tip filtriranja je tudi iskanje po paketih. Naša aplikacija omogoča iskanje po dveh kriterijih:

- glede na zapis informacij o paketu (*by Info*)
- glede na zapis celotnega paketa v bajtih (*by Bytes*)

Iskanje glede na zapis informacij o paketu poteka tako, da uporabnik v polje (slika 4.16) vnese željeni niz. Na zaslon se izpišejo paketi, pri katerih je ta niz podniz atributa `Info`. Podobno velja za iskanje glede na zapis celotnega paketa v bajtih, le da mora biti tokrat iskani niz podniz atributa `All`. Pod ta atribut se shrani paket v obliki `XX-XX-...-XX-XX`, torej s pomišljaji med bajti. V enakem formatu mora uporabnik vnesti iskani podniz bajtov.



Slika 4.16: Iskanje po paketih.

Pri implementaciji iskanja smo uporabili tudi števec (`DispatcherTimer` iz imenskega prostora `System.Windows.Threading`). Z njim si pomagamo, ko želimo, da med pisanjem iskanje glede na napisano ne poteka za vsak spremenjen znak. S tem precej pospešimo delovanje iskanja. Ko se zazna sprememba (nov znak ali izbrisan znak) v polju za iskanje, se števec požene. Če ni nove spremembe v naslednjih 800 milisekundah, potem filtriramo pakete glede na vnešeni niz. Če pa pride do spremembe, se števec resetira.

Ko filtriramo pakete, velja med filtri naslednja logika:
(Tip paketa **in** Oznaka vrat **ali** FN Error **ali** XMC Error**) in** Iskanje po Info **in** Iskanje po bajtih.

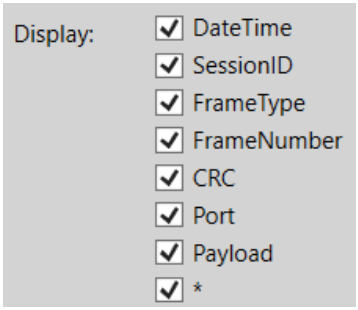
To pomeni, da v kolikor vnesemo niz v katerega izmed iskalnikov, se bodo zagotovo izločili vsi paketi, ki temu nizu ne bodo ustrezali. Če pa odznačimo vse tipe paketov in vrat, obkljukani pa imamo obe specifični napaki, se bodo vseeno na izpisu pojavili paketi, pri katerih je označena ta napaka, čeprav imajo seveda določen tip.

V programski kodi je ta logika predstavljena kot `if` pogoj. Če nek paket (objekt razreda `FrameModel`) ustreza temu pogoju, se doda v seznam za izpis na zaslonu. V kolikor je paket tipa App/Data, se še prej preveri, če ustreza izbranemu filtru glede na XMC podatke (*All*, *Only XMC Data* ali *W/o XMC Data*).

4.6 Izbira prikaza stolpcev

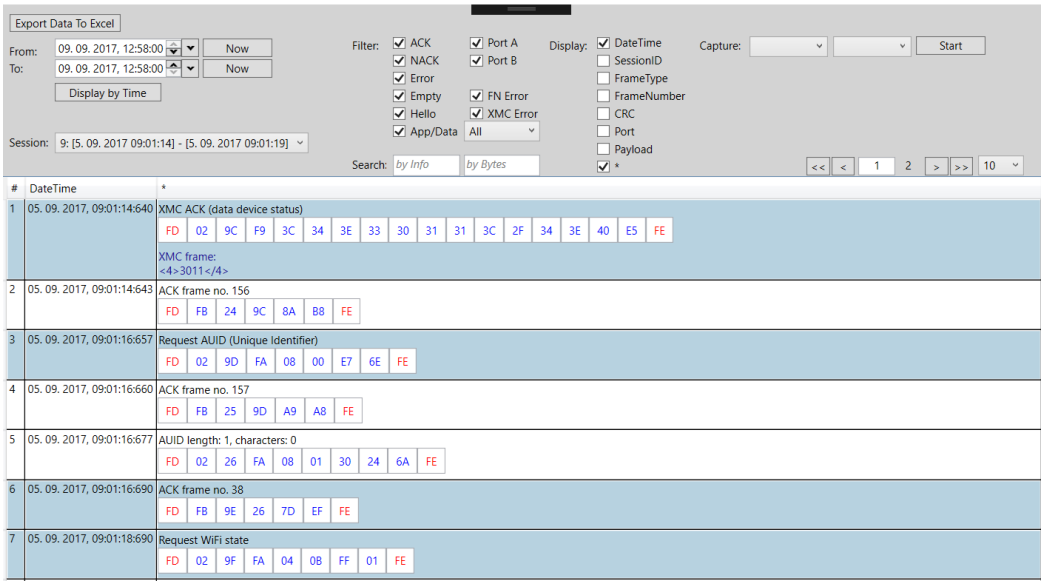
Včasih uporabnik nekaterih podatkov o paketu ne potrebuje in ga zanima zgolj določena lastnost. Na primer, če išče napako med potrditvami paketov,

potem ga ne zanimajo kontrolni biti (CRC) in želi stolpec s podatki o njih skriti. Zato smo uporabniku omogočili, da si lahko izbere, katere stolpce z informacijami o paketu želi prikazati na zaslon. Slika 4.17 prikazuje polje, pri katerem uporabnik izbere, katere stolpce želi videti:



Slika 4.17: Polje za izbiro prikaza stolpcev.

Na sliki 4.18 vidimo izpis paketov v primeru, ko izklopimo določene stolpce. V tem primeru imamo vklopljen le DateTime in *, # pa se vedno izpiše.

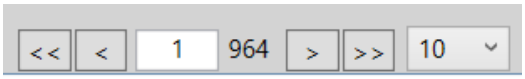


Slika 4.18: Izpis paketov z deaktiviranimi stolpci.

4.7 Ostranjevanje

V zbirki paketov, ki jih želi uporabnik izpisati na zaslon, je lahko tisoče ali celo milijone paketov, kar je preobsežno za izpis na eno stran. Zato ima aplikacija implementirano ostranjevanje (ang. *paging*), pri čemer ima uporabnik na voljo več možnih izbir, koliko zadetkov na stran želi prikazati. Izbira lahko med 10, 20, 50, 100 ali 1000 zadetkov na stran.

Za sprehajanje po straneh imamo 4 gumbe - na eni strani za premik na prvo ter na prejšnjo stran, na drugi strani pa za premik na naslednjo in za premik na zadnjo stran. Vmes je vnosno polje, ki kaže, na kateri strani smo, ter oznaka, ki pove, koliko je vseh strani. Slika 4.19 prikazuje prikaz polja, s katerim se uporabnik sprehaja po straneh.



Slika 4.19: Polje za ostranjevanje.

Uporabnik lahko v vnosno poljo sam vnese številko strani, ki jo želi prikazati. Pri vnosu se preverja, če je ta veljaven - neveljaven je v primeru, če znak, ki ga vnese, ni številka, oziroma če je številka izven obsega strani (od 1 do števila vseh strani). Če je vnos neveljaven, se na zaslonu prikaže okno z opozorilom o napaki.

Število vseh strani dobimo po naslednji formuli:

$$\left\lceil \frac{\#paketov\ za\ izpis}{\#izbranih\ zadetkov\ na\ stran} \right\rceil$$

Opomba: *znak '#' predstavlja število.*

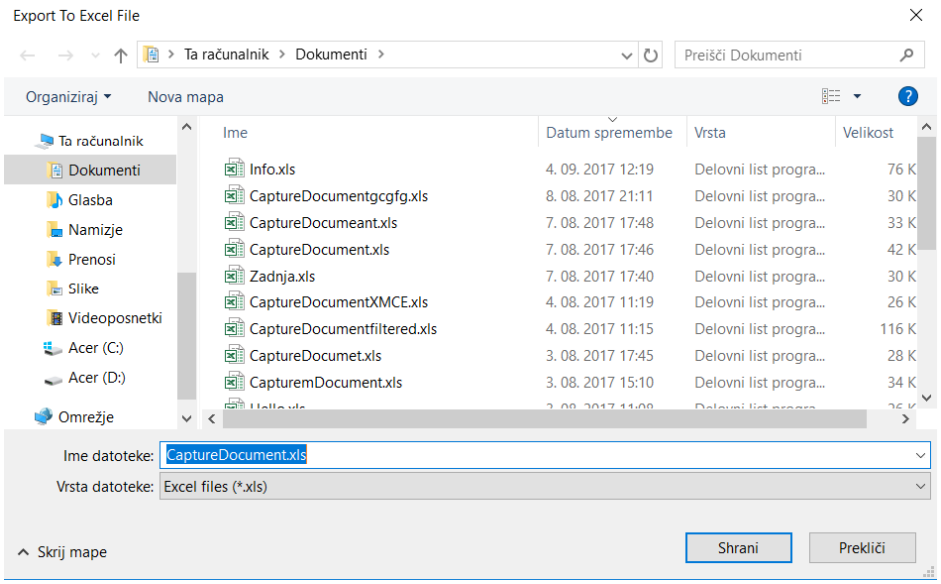
Vsak objekt razreda `FrameModel`, torej vsak paket, ki ga želi uporabnik izpisati, nosi podatek o tem, na kateri strani se nahaja. To pove atribut `Page`. Ob spremembi paketov za izpis ali ob spremembi števila izpisanih zadetkov na stran ta atribut nastavimo na novo. Ob izpisu določene strani

preverimo, kateri paketi imajo `Page` nastavljen na izbrano stran. Ti paketi se prikažejo na zaslonu.

4.8 Izvoz izpisa v Excelov dokument

Uporabnik aplikacije bo morda želel prestrežene pakete natisniti, jih komu poslati, dodati opombe ali jih kako drugače obdelati oziroma uporabiti. Za to potrebujemo izvoz izpisa paketov v Excelov dokument.

Če želi uporabnik izvoziti izpis paketov v Excelov dokument, to naredi s klikom na gumb "Export Data To Excel". Nato se mu odpre okno, kot ga prikazuje slika 4.20.

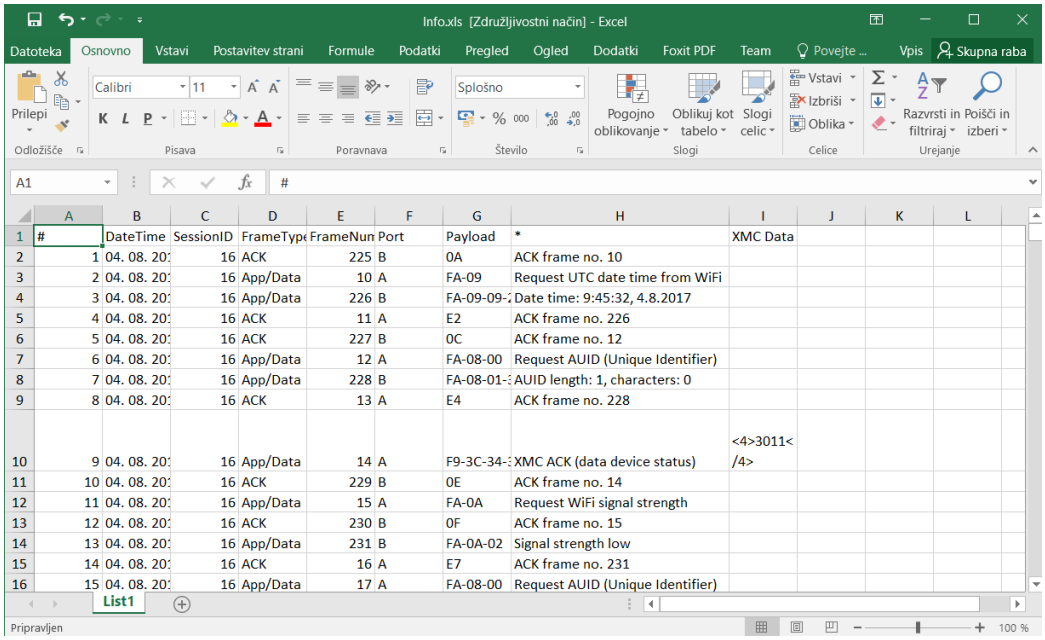


Slika 4.20: Okno za izbiro lokacije dokumenta.

V tem oknu izbere lokacijo, kamor želi shraniti izvožen Excelov dokument, ter vnese njegovo ime. To je na začetku ponujeno kot "CaptureDocument.xls".

Če se med samim izvozom izpisa paketov postopek predčasno zaključi (pride do napake), se uporabniku na zaslonu prikaže okno z opozorilom, da izvoz ni bil uspešen.

V dokument se izvozijo paketi, ki jih uporabnik izbere za prikaz. Pri tem so seveda upoštevani tudi morebitni vključeni filtri. Podatki o paketih se izpišejo v 9 stolpcev: #, DateTime, SessionID, FrameType, FrameNumber, Port, Payload, * in XMC Data. Prvih 7 stolpcev je enakih kot pri izpisu paketov na zaslon, medtem ko pod * izpisujemo le informacije o paketu (`Info`), pod XMC Data pa morebitne podatke XMC protokola. Teh 9 stolpcev se izpiše v vsakem primeru, ne glede na uporabnikovo izbiro prikaza stolpcev v aplikaciji. Primer izvoženega Excelovega dokumenta prikazuje slika 4.21.



#	DateTime	SessionID	FrameType	FrameNum	Port	Payload	*	XMC Data
1	04. 08. 20	16	ACK	225	B	0A	ACK frame no. 10	
2	04. 08. 20	16	App/Data	10	A	FA-09	Request UTC date time from WiFi	
3	04. 08. 20	16	App/Data	226	B	FA-09-09-	Date time: 9:45:32, 4.8.2017	
4	04. 08. 20	16	ACK	11	A	E2	ACK frame no. 226	
5	04. 08. 20	16	ACK	227	B	0C	ACK frame no. 12	
6	04. 08. 20	16	App/Data	12	A	FA-08-00	Request AUID (Unique Identifier)	
7	04. 08. 20	16	App/Data	228	B	FA-08-01-	AUID length: 1, characters: 0	
8	04. 08. 20	16	ACK	13	A	E4	ACK frame no. 228	
9	04. 08. 20	16	App/Data	14	A	F9-3C-34-	XMC ACK (data device status)	<4>3011</4>
10	04. 08. 20	16	ACK	229	B	0E	ACK frame no. 14	
11	04. 08. 20	16	App/Data	15	A	FA-0A	Request WiFi signal strength	
12	04. 08. 20	16	ACK	230	B	0F	ACK frame no. 15	
13	04. 08. 20	16	App/Data	231	B	FA-0A-02	Signal strength low	
14	04. 08. 20	16	ACK	16	A	E7	ACK frame no. 231	
15	04. 08. 20	16	App/Data	17	A	FA-08-00	Request AUID (Unique Identifier)	

Slika 4.21: Izpis paketov v Excelovem dokumentu.

Pri implementaciji izvoza izpisa paketov v Excelov dokument smo si pomagali z razredom `SaveFileDialog` iz imenskega prostora `Microsoft.Win32`. Ta poskrbi, da se odpre okno, kjer uporabnik izbere lokacijo shranjevanja in ime dokumenta (slika 4.20). Samo pretvorbo izpisa paketov implementiramo s pomočjo razredov in funkcij iz imenskega prostora `Microsoft.Office.Interop.Excel`.

Poglavje 5

Zaključek ter možne nadgradnje aplikacije

V okviru te diplomske naloge smo razvili aplikacijo, s katero lahko prestrezamo komunikacijo med WiFi modulom in elektroniko gospodinjskega aparata. Ta poteka v obliki paketov protokola Inspector. Te pakete lahko uporabnik aplikacije izpiše na zaslon, jih filtrira, išče določene napake, lahko pa jih tudi izvozi v Excelov dokument.

Aplikacijo smo ves čas razvijali z mislijo na nadaljni razvoj. Pri prikazu paketov imamo možnost, da s klikom izberemo določen bajt. To nam bo prav prišlo, ko bomo želeli paket še bolj razčleniti in povezati vsak bajt z njegovim pomenom. V prihodnje želimo tudi dodati, da bi program zaznal popolnoma popačene pakete, torej takšne, ki imajo poškodvan zečetek paketa (SOF) ali konec paketa (EOF) in jih zato ni mogoče zaznati.

Ker se lahko pri problemu prestrezanja paketov soočamo z veliko količino podatkov, bi bila dobrodošla večja optimizacija aplikacije. Na primer pri dostopanju do baze, pri izpisu paketov, ostranjevanju, filtriranju in iskanju po paketih ter pri poslušanju "v živo".

Trenutna aplikacija ne razčleni XMC vsebine. To bi lahko bilo koristno, saj nam ta pove, kaj se dogaja z gospodinjskim aparatom.

Če bi imeli implementirano razčlembo XMC protokola, bi lahko pakete s

temi podatki prestrezali tudi v samem omrežju WiFi. Za to niti ne bi potrebovali nove aplikacije, ampak bi lahko ustvarili modul za program Wireshark ali za kakšen drug program, ki je specializiran za prestrezanje paketov v omrežju.

Literatura

[1] Domotika, pametna hiša. Dosegljivo: http://kske.fgg.uni-lj.si/award/html/2/0_0.html#2.0. [Dostopano: 18. 8. 2017].

[2] Amazon alexa. Dosegljivo: https://en.wikipedia.org/wiki/Amazon_Alexa. [Dostopano: 20. 8. 2017].

[3] Google assistant. Dosegljivo: https://en.wikipedia.org/wiki/Google_Assistant. [Dostopano: 20. 8. 2017].

[4] Ge connected appliances. Dosegljivo: http://www.geappliances.com/ge/connected-appliances/. [Dostopano: 20. 8. 2017].

[5] Home connect. Dosegljivo: http://www.home-connect.com/de/en/. [Dostopano: 20. 8. 2017].

[6] Ifa 2016 - gorenje. Dosegljivo: http://www.gorenjegroup.com/si/za-medije/medijske-knjiznice/ifa-2016/asko. [Dostopano: 1. 9. 2017].

[7] Nest. Dosegljivo: https://nest.com. [Dostopano: 21. 8. 2017].

[8] Nest announces open source implementation of thread. Dosegljivo: https://nest.com/at/press/nest-announces-open-source-implementation-of-thread/. [Dostopano: 21. 8. 2017].

[9] 11 internet of things (iot) protocols you need to know about. Dosegljivo: https://www.rs-online.com/designspark/eleven-

internet-of-things-iot-protocols-you-need-to-know-about. [Dostopano: 21. 8. 2017].

[10] Home automation protocols: A round-up. Dosegljivo: https://www.electronichouse.com/smart-home/home-automation-protocols-what-technology-is-right-for-you/. [Dostopano: 21. 8. 2017].

[11] Home automation. Dosegljivo: https://en.wikipedia.org/wiki/Home_automation. [Dostopano: 19. 8. 2017].

[12] Zigbee. Dosegljivo: https://en.wikipedia.org/wiki/Zigbee. [Dostopano: 23. 8. 2017].

[13] Serial port. Dosegljivo: https://en.wikipedia.org/wiki/Serial_port. [Dostopano: 28. 8. 2017].

[14] Pan9320 - product specification. Dosegljivo: https://na.industrial.panasonic.com/sites/default/pidsa/files/downloads/files/panasonic-pan9310-9320-datasheet.pdf. [Dostopano: 29. 8. 2017].

[15] Ge connectplus - slika. Dosegljivo: https://www.myapstore.com/MarketingObjectRetrieval/Dispatcher?RequestType=Image&Name=PBX10W00Y0-9838.jpg. [Dostopano: 10. 9. 2017].

[16] Amazon echo - slika. Dosegljivo: https://images-na.ssl-images-amazon.com/images/I/41-v1fozy0L.jpg. [Dostopano: 22. 8. 2017].

[17] Google home - slika. Dosegljivo: https://storage.googleapis.com/madebygoog/v1/banners/home_banner.jpg. [Dostopano: 22. 8. 2017].

[18] Vrednost svetovnega trga domotike. Dosegljivo: https://web.archive.org/web/20160505124414/https:

//www.reuters.com/article/research-and-markets-idUSnBw195490a%2B100%2BBSW20150119. [Dostopano: 19. 8. 2017].

[19] The story of information. Dosegljivo: https://infostory.com/2010/09/16/the-first-home-computer-1965/. [Dostopano: 19. 8. 2017].

[20] The history of smart homes. Dosegljivo: http://www.iotevolutionworld.com/m2m/articles/376816-history-smart-homes.htm. [Dostopano: 19. 8. 2017].

[21] The most powerful internet of things companies. Dosegljivo: http://www.computerworlduk.com/galleries/data/most-powerful-internet-of-things-companies-3521713/. [Dostopano: 20. 8. 2017].

[22] Voice activated appliances. Dosegljivo: http://www.geappliances.com/ge/connected-appliances/voice-activated-appliances.htm. [Dostopano: 20. 8. 2017].

[23] Branko Šter. Računalniška vrata (porti). Dosegljivo: https://ucilnica.fri.uni-lj.si/pluginfile.php/54465/mod_resource/content/2/RS-7-porti.pdf. [Dostopano: 29. 8. 2017].